ECE482 — Introduction to Operating Systems

Lab 4

Manuel — JI (Fall 2024)

Goals of the lab

- Learn basics on database
- Import and run simple database queries
- Learn basics on debugging with GDB

ECE4820 Tasks

1 Database

This is the evening, you are exhausted after a long day of work on `mumsh`. So you decide to poke around and learn more about database, as unfortunately you never had to opportunity to select such course during your studies.

1.1 Database creation

As a first step you need to find a database, you fire-up your web-browser. Unfortunately your internet is very slow today so you cannot get much information. But after a bit of thinking you realise that you still have a git version of the Linux kernel, and as you know everything about git you can easily generate logs from the git commits. To ensure a proper formatting you refer to git pretty format documentation page.

So you open a terminal running `mumsh` and type a simple command line to test database generation.

```
mumsh $ git log --pretty="%H,%aN,%aI,%s" > db.csv
```

The goal being to get a basic introduction to database you only want to focus on basic queries, in particular you do not need a very complicated database and in the end only generating two csv files[1] containing the following fields is enough.

Fields for `timestamp.csv`:

- Hash of the commit
- Author name
- Author date, strict ISO 8601 format
- Author date, UNIX timestamp

Fields for `db.csv`:

- Hash of the commit
- Author name
- Subject

1.2 Database system installation

As you want to ensure your understanding and guesses are correct you need to verify a few things online. Luckily your network seems back to normal, so you can use a proper search engine and ensure the correctness of what you found.

- What are the most common database systems?

[1]The `db.csv` and `timestamp.csv` can be found on the server in the directory `ve482/l4`.

- Briefly list the pros and cons of the three most common ones.

After completing your reading you decide to install `SQLite` on your Linux system. The next step is now to import your git database into two tables.

- Create an empty SQLite database.
- Use the SQLite shell to prepare two empty tables for each of your .csv file.
- Import each .csv file in its corresponding SQLite table.

1.3 Database queries

At this stage you want to run basic queries to verify that the database has been imported correctly. Therefore you spend the rest of the evening playing around the database and running queries.

- Who are the top five contributors to the Linux kernel since the beginning?
- Who are the top five contributors to the Linux kernel for each year over the past five years?
- What is the most common "commit subject"?
- On which day is the number of commits the highest?
- Determine the average time between two commits for the five main contributors.

2 Debugging

You are pretty happy and enjoying the database tasks when your mum pops in your room. She looks pretty upset that you are still not asleep as she thinks you were playing video games...

When you explain her to that you have terrible bugs in your shell and needed a bit of change, she asked you whether you had used GDB. As you replied "Can I eat it?" she realises you probably do not know much about it. She kindly tells you to have a quick try at it on your current `mumsh` version to preview it. This should become very handy if you ever have to work on a large scale project.

1. How to enable built-in debugging in `gcc`?
2. What is the meaning of GDB?
3. Compile the master branch of you `mumsh` with debugging enabled.

2.1 Basic GDB usage

1. Find the homepage of the GDB project.
2. What languages are supported by GDB?

Mum who is now fully awaken kindly guides you in your discovery of GDB. In a terminal navigate to the folder where the previously mentioned program is located. Enter `gdb` in the terminal. An interactive shell should open, under the condition that GDB has been installed. First note that this shell recalls history when using the arrow keys and auto-completes command using the `TAB` key.

At any stage `help [command]` can be typed to get general information or details on a specific command. To debug the program `mumsh` input `file mumsh`. The program is now loaded but not run. In order to run it, input the command `run`.

The command `break mumsh.c:17` sets a breakpoint at line 17 in the file `mumsh.c`. Upon executing `run`, the program pauses each time it encounters a breakpoint. There is no restriction on the number of breakpoints added to a program.

An alternative strategy consists in breaking at a certain function. In that case input for instance `break function_name`, in order to stop each time the function `function_name` is called.

When blocked at a breakpoint computation can be resumed using the command `continue` but will stop at the next breakpoint. In order to progress line-by-line use the command `step`. If more than one instruction is written on a line input `next` to only run a single instruction at a time. Since typing in `next` or `step` many time can be tedious GDB offers the possibility the repeat the previous command by just pushing the `ENTER` key.

In order for debugging to be effective it is necessary to observe things at breakpoints: use `print tmp` to display the value of the variable `tmp`. Special breakpoints, called watchpoints, can be used to track a variable. This is achieved through the `watch tmp` command. Each time `tmp` is modified the program stops and display both its old and new values.

For a pointer `p`, `print p` prints the memory address of the pointer. Accessing each element of a structure is done in a similar way as in C, e.g. `p->s` displays the field `s` referenced by pointer `p`. Note that it is also possible to input `print *p` even if `p` is a pointer on a structure (this is not easily done in C).

ECE4821 Tasks

3. What are the following GDB commands doing:
   - `backtrace`
   - `where`
   - `finish`
   - `delete`
   - `info breakpoints`
4. Search the documentation and explain how to use conditional breakpoints.

Watch this youtube video and answer the following questions.

5. What is `-tui` option for GDB?
6. What is the “reverse step” in GDB and how to enable it. Provide the key steps and commands.

2.2 Basic LLDB usage

Although nice and helpful, GDB has some limitations which are addressed with the development of LLDB. In particular among the most important goals of LLDB are the proper handling of complex expression parsing, overloading, templates, and multithreading.

- Quickly go through `https://lldb.llvm.org/use/map.html` to understand how to transition from `gdb` to LLDB;
- Follow LLDB tutorial presented at `https://lldb.llvm.org/use/tutorial.html`;